EECS151/251A
Spring 2018
Digital Design and Integrated Circuits

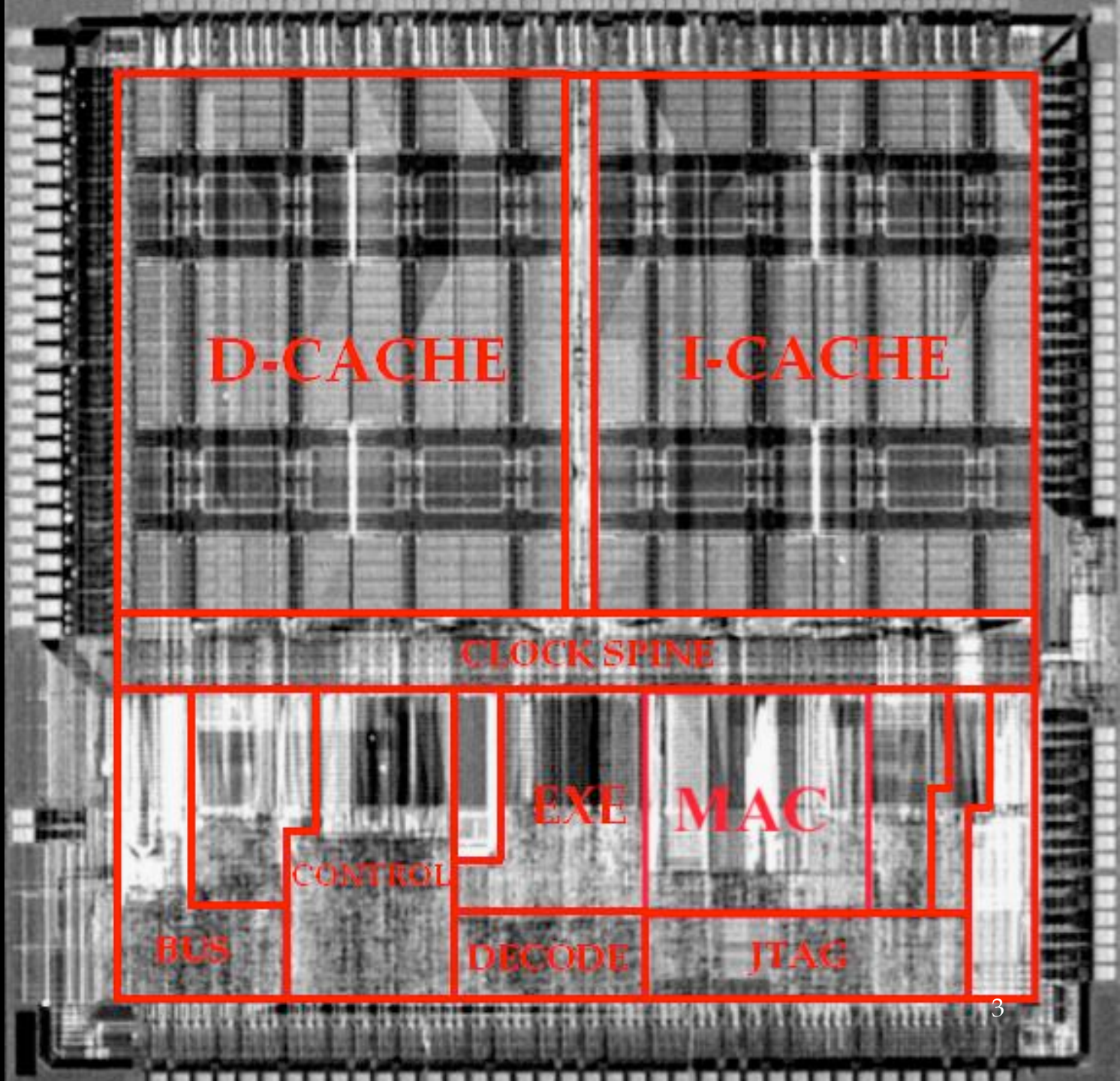
Instructors:
John Wawrzynek and Nick Weaver

Lecture 19: Caches

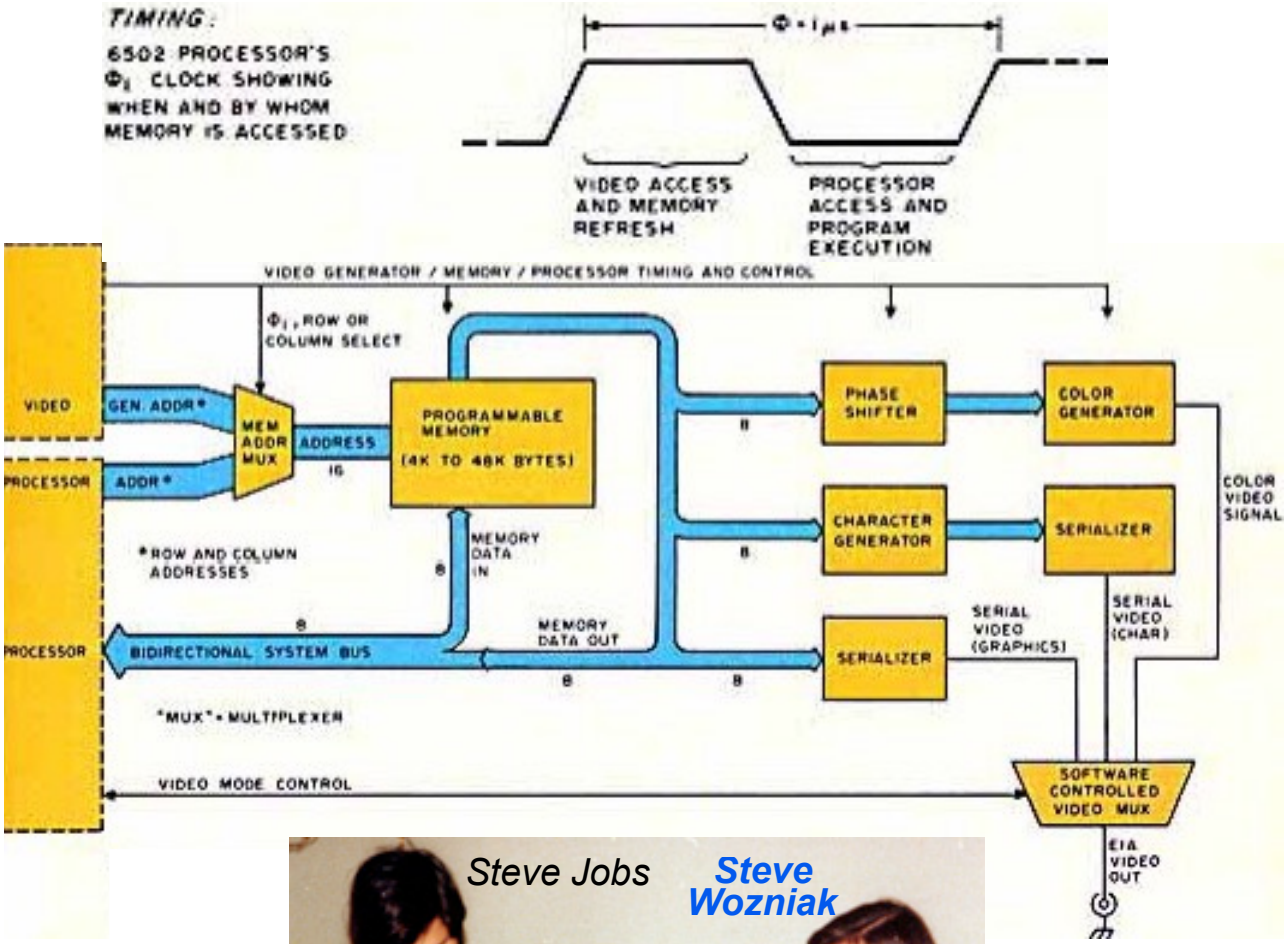
Cache Introduction

40% of this ARM CPU is devoted to SRAM cache.

But the role of cache in computer design has varied widely over time.



1977: DRAM faster than microprocessors

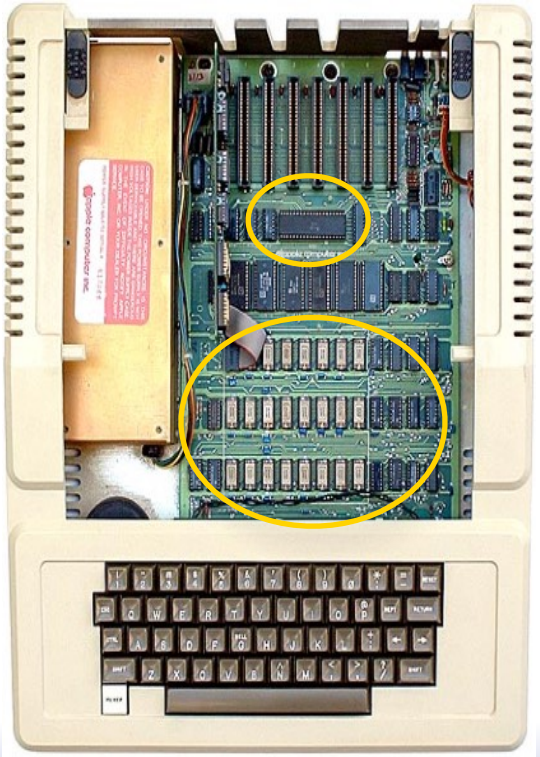


RAM Complement	Apple II System
4K	$ 1,298.00
48K	2,638.00

Apple II (1977)

CPU: 1000 ns

DRAM: 400 ns

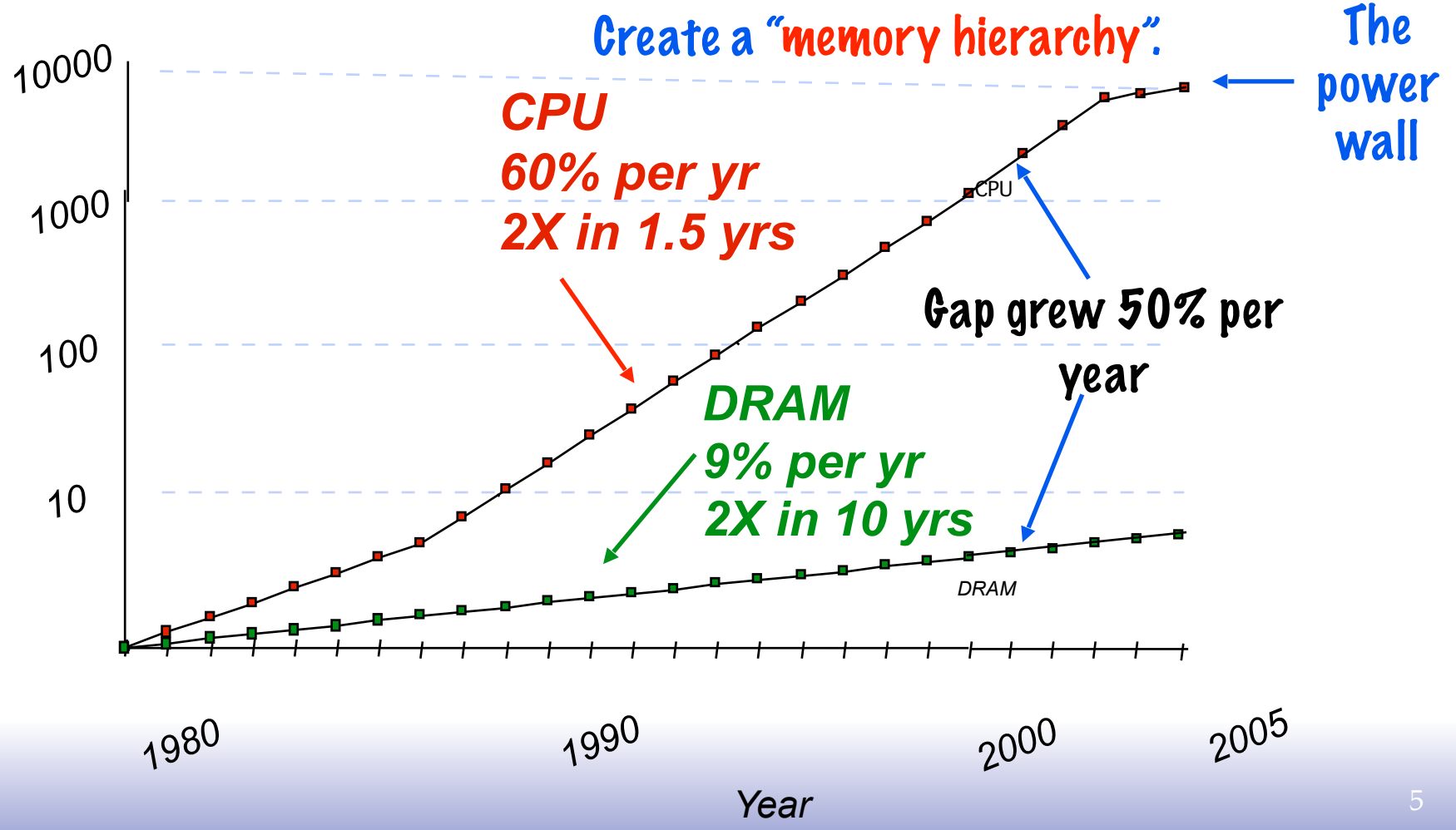


1980-2003, CPU speed outpaced DRAM ...

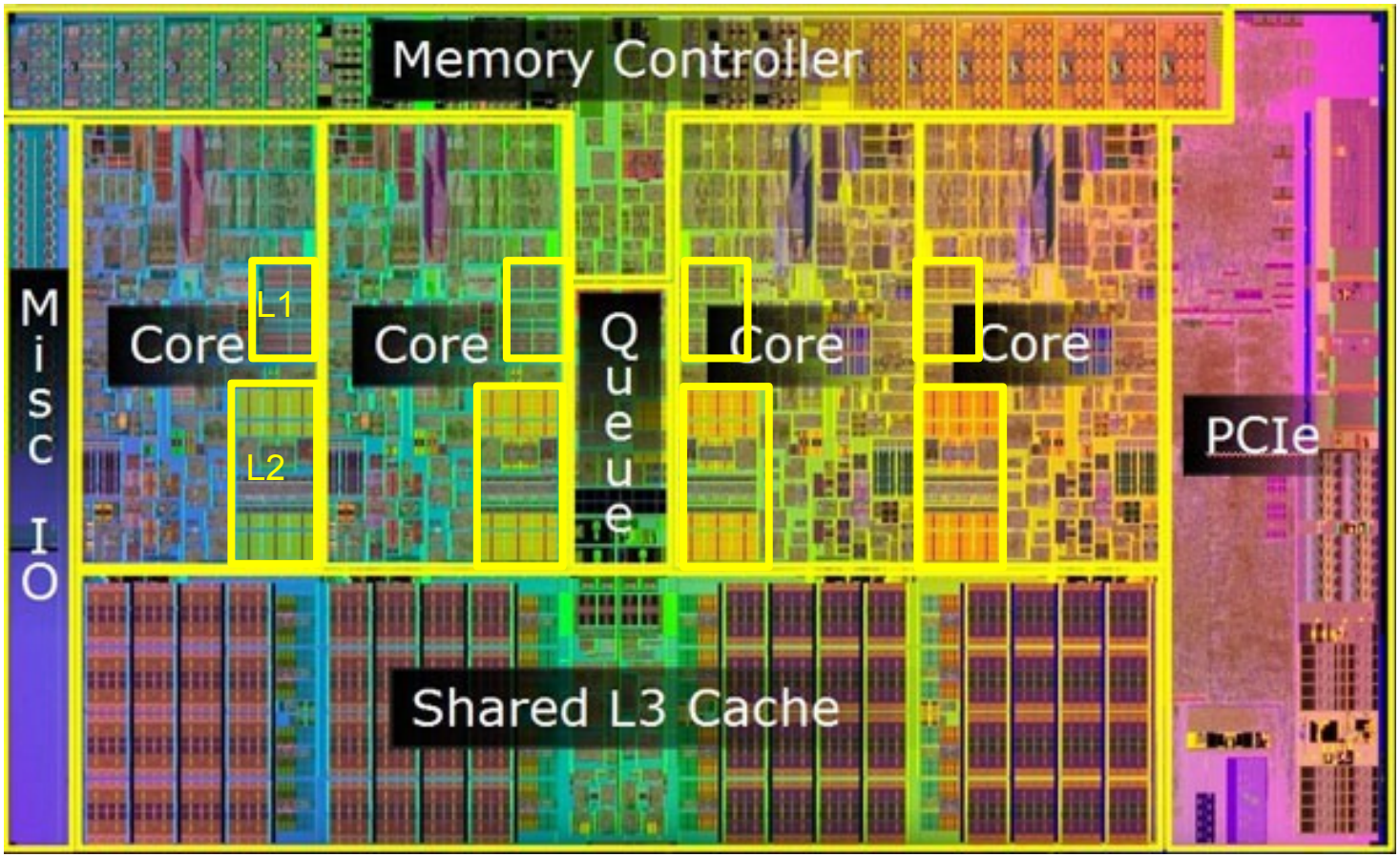
Q. How do architects address this gap?

A. Put smaller, faster "cache" memories between CPU and DRAM.

Create a "memory hierarchy".



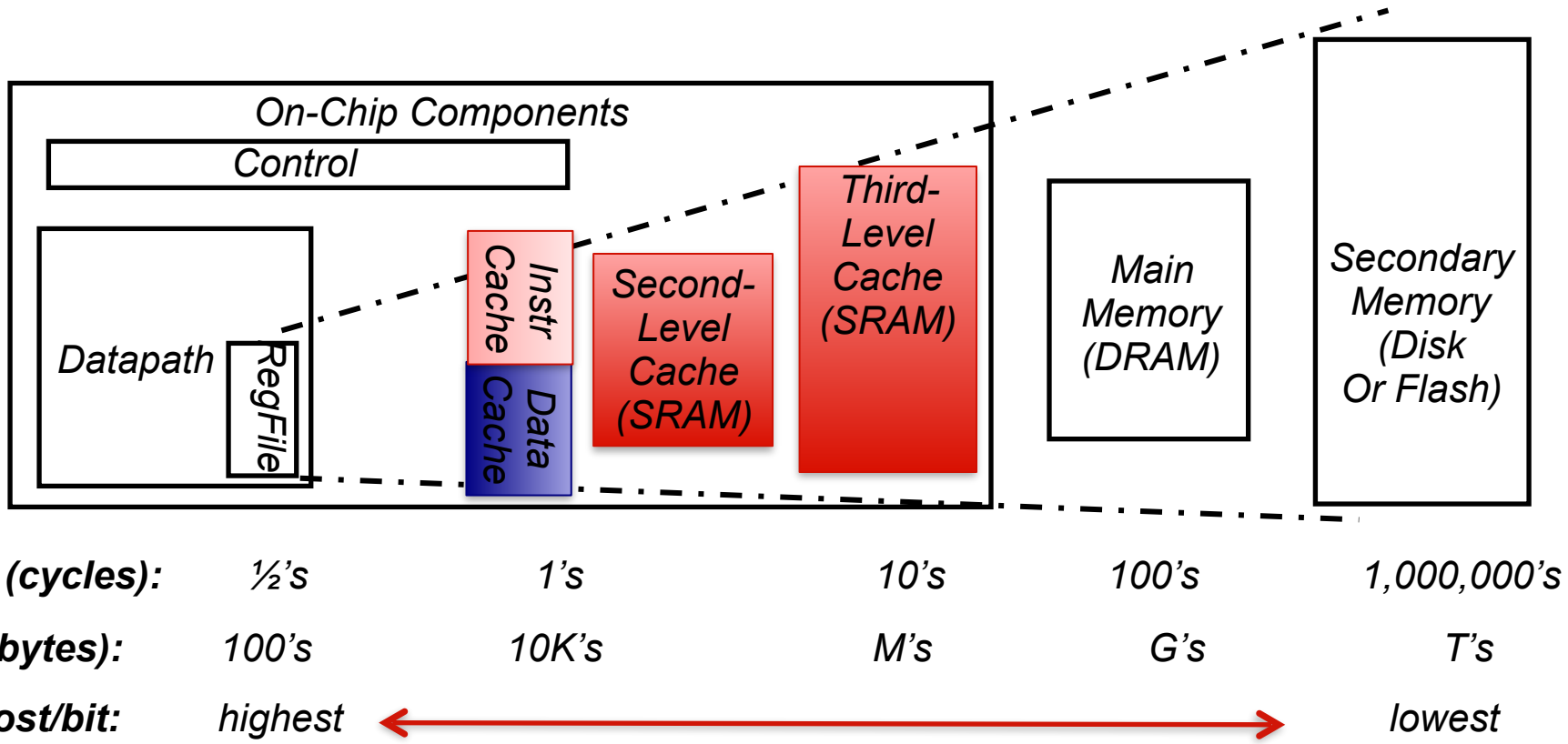
Nahalem Die Photo (i7, i5)



- Per core:
  - 32KB L1 I-Cache (4-way set associative)
  - 32KB L1 D-Cache (8-way set associative)
  - 256KB unified L2 (8-way SA, 64B blocks)
- Common L3 8MB cache

Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

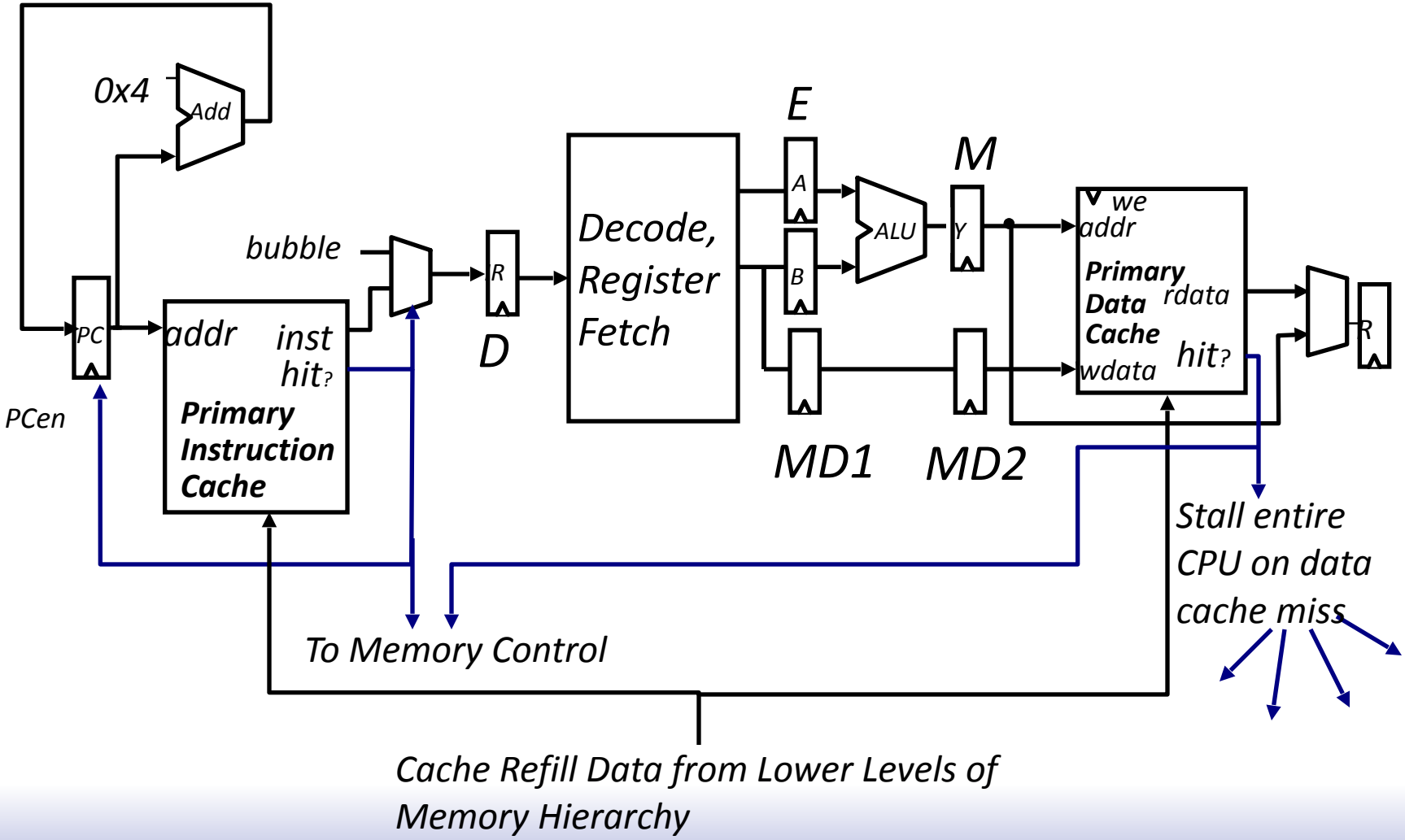
Typical Memory Hierarchy



- Principle of locality + memory hierarchy presents programmer with ≈ as much memory as is available in the *cheapest* technology at the ≈ speed offered by the *fastest* technology

CPU-Cache Interaction

(5-stage pipeline)



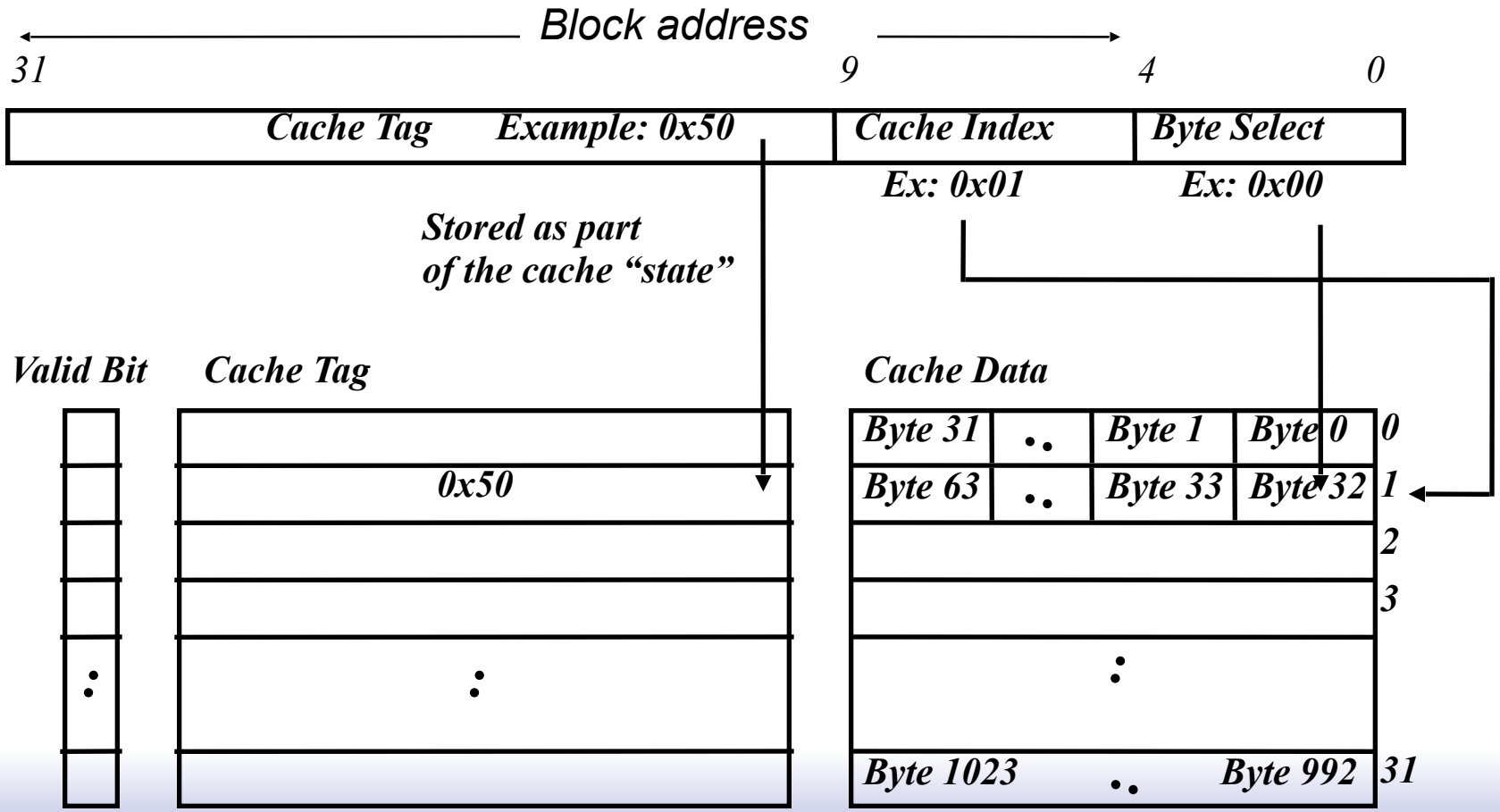
Review from 61C

- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.
- DRAM is slow but cheap and dense:
  - Good choice for presenting the user with a BIG memory system
- SRAM is fast but expensive and not very dense:
  - Good choice for providing the user FAST access time.

Example: 1 KB Direct Mapped Cache with 32 B Blocks

For a 2^N byte cache:

- The uppermost (32 - N) bits are always the Cache Tag
- The lowest M bits are the Byte Select (Block Size = 2^M)

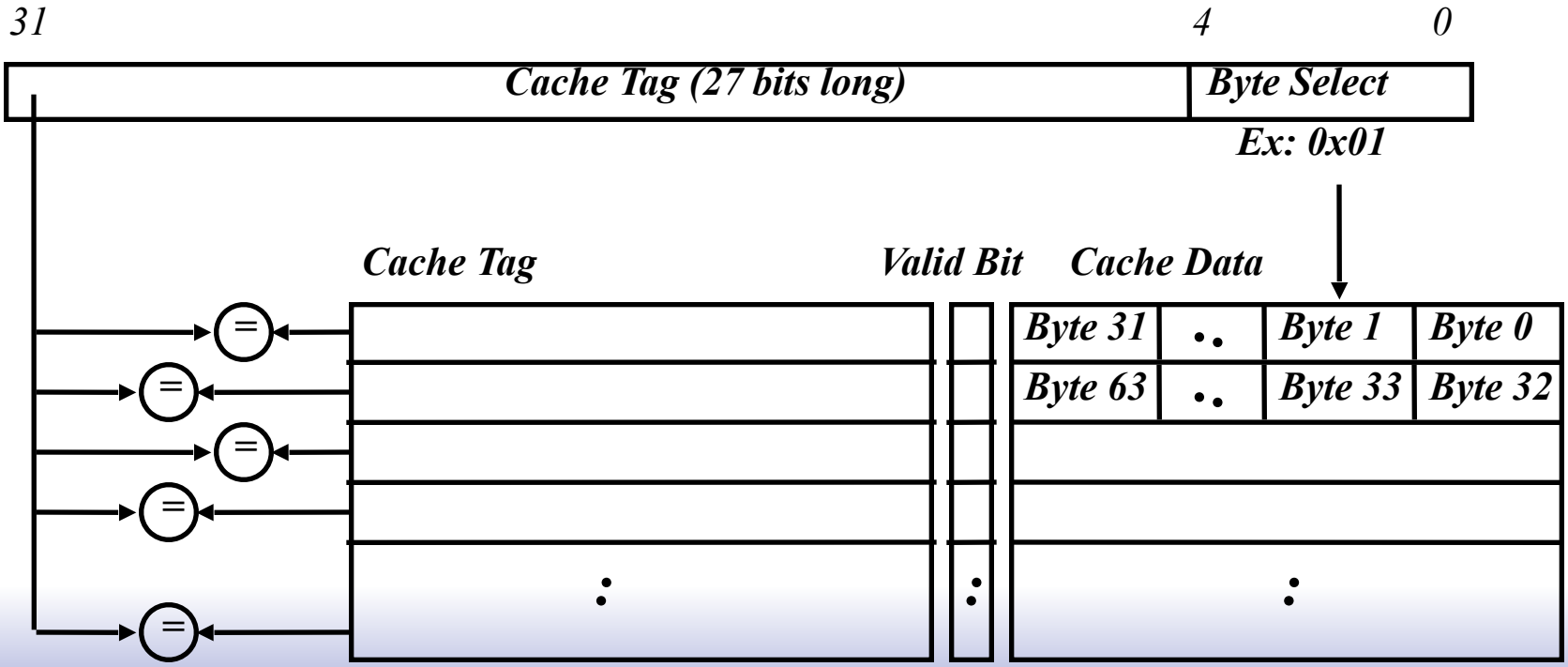


Extreme Example: Fully Associative

Fully Associative Cache

- Forget about the Cache Index
- Compare the Cache Tags of all cache entries in parallel
- Example: Block Size = 32 B blocks, we need N 27-bit comparators

By definition: Conflict Miss = 0 for a fully associative cache



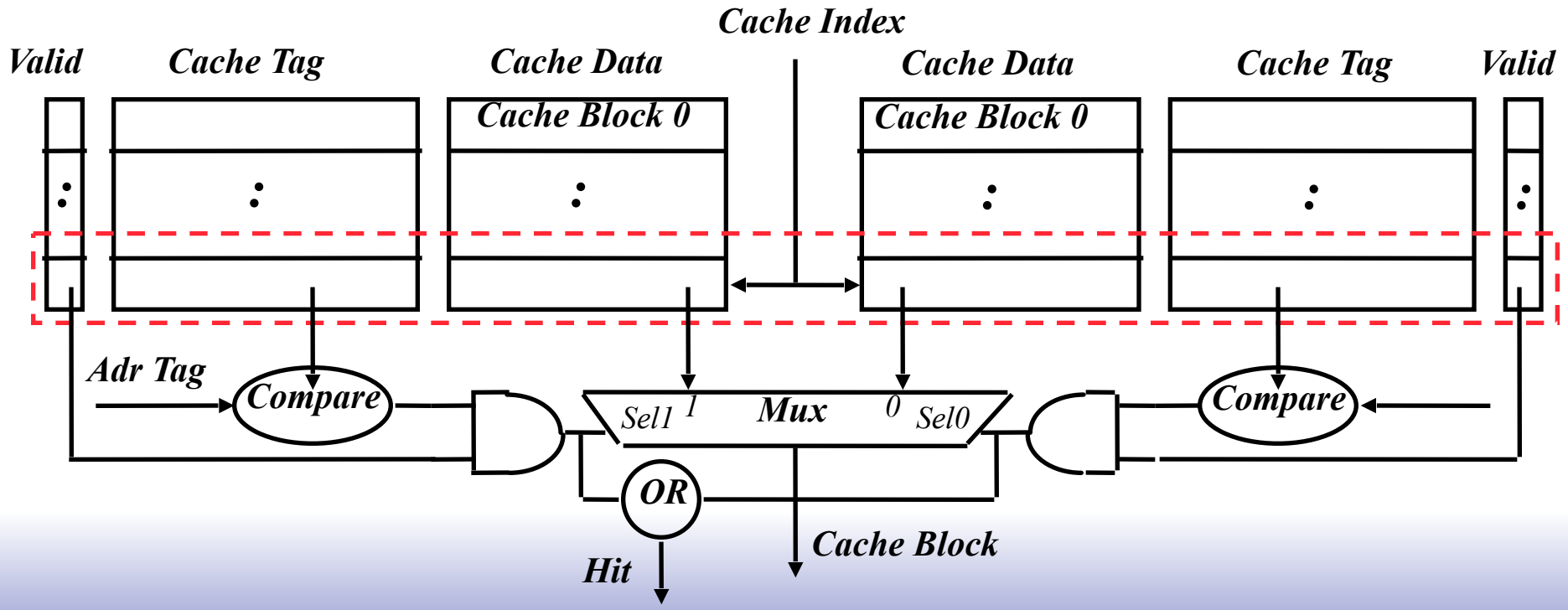
Set Associative Cache

N-way set associative: N entries for each Cache Index

- N direct mapped caches operates in parallel

Example: Two-way set associative cache

- Cache Index selects a "set" from the cache
- The two tags in the set are compared to the input in parallel
- Data is selected based on the tag result



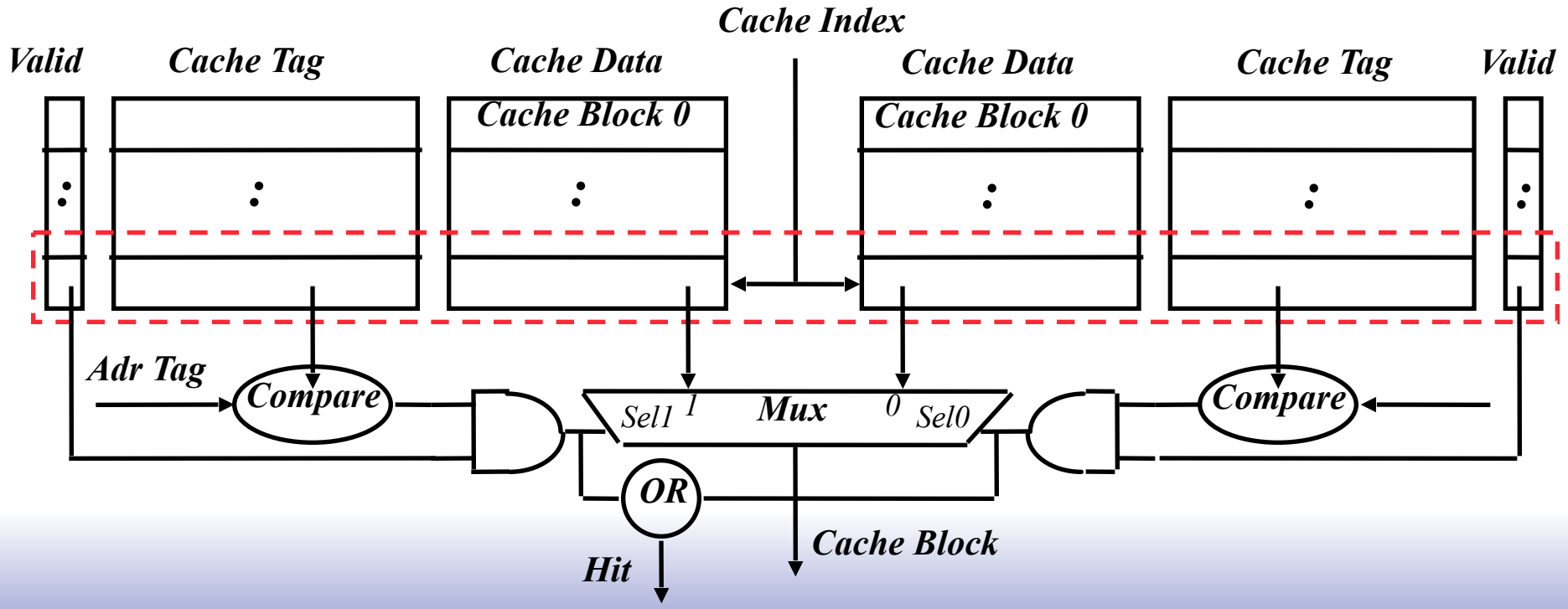
Disadvantage of Set Associative Cache

N-way Set Associative Cache versus Direct Mapped Cache:

- N comparators vs. 1
- Extra MUX delay for the data
- Data comes AFTER Hit/Miss decision and set selection

In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:

- Possible to assume a hit and continue. Recover later if miss.



Cache Design and Optimization

Performance =
Intr. Count x Clock Freq x (ideal CPI + stalls)

Average Memory Access time =
Hit Time + Miss Rate x Miss Penalty

Improving Cache Performance: 3 general options

1. **Reduce the miss rate,**
2. **Reduce the miss penalty, or**
3. **Reduce the time to hit in the cache (although this is very often 1 cycle).**

Primary Cache Parameters

- Block size
  - how many bytes of data in each cache entry?
- Associativity
  - how many ways in each set?
  - Direct-mapped => Associativity = 1
  - Set-associative => 1 < Associativity < #Entries
  - Fully associative => Associativity = #Entries
- Capacity (bytes) = Total #Entries * Block size
- #Entries = #Sets * Associativity

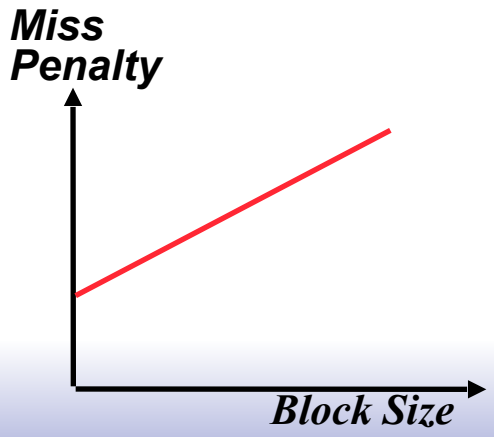
Block Size Tradeoff

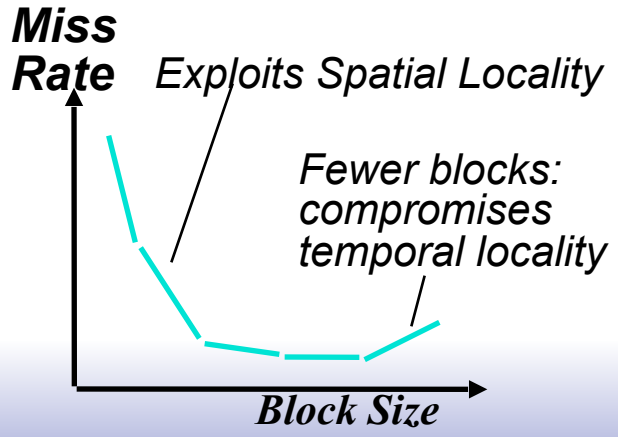
In general, larger block size takes advantage of spatial locality **BUT**:

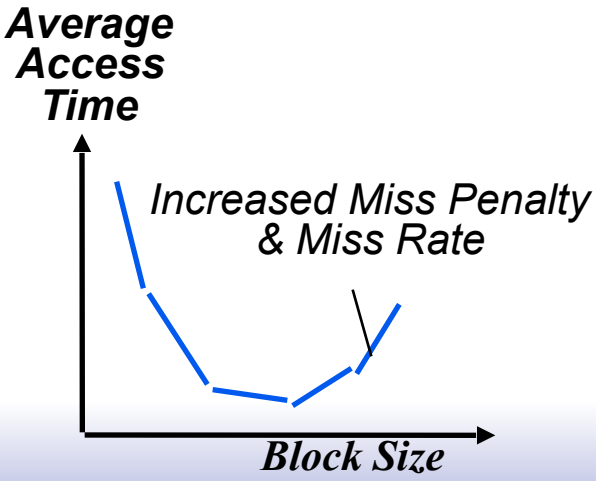
- Larger block size means larger miss penalty:
  - Takes longer time to fill up the block
- If block size is too big relative to cache size, miss rate will go up
  - Too few cache blocks

In general, Average Memory Access Time (AMAT):

= Hit Time + Miss Penalty x Miss Rate







Summary on Sources of Cache Misses

Compulsory (cold start, first reference): first access to a block

- "Cold" fact of life: not a whole lot you can do about it
- Note: If you are going to run "billions" of instruction, Compulsory Misses are insignificant

Conflict (collision):

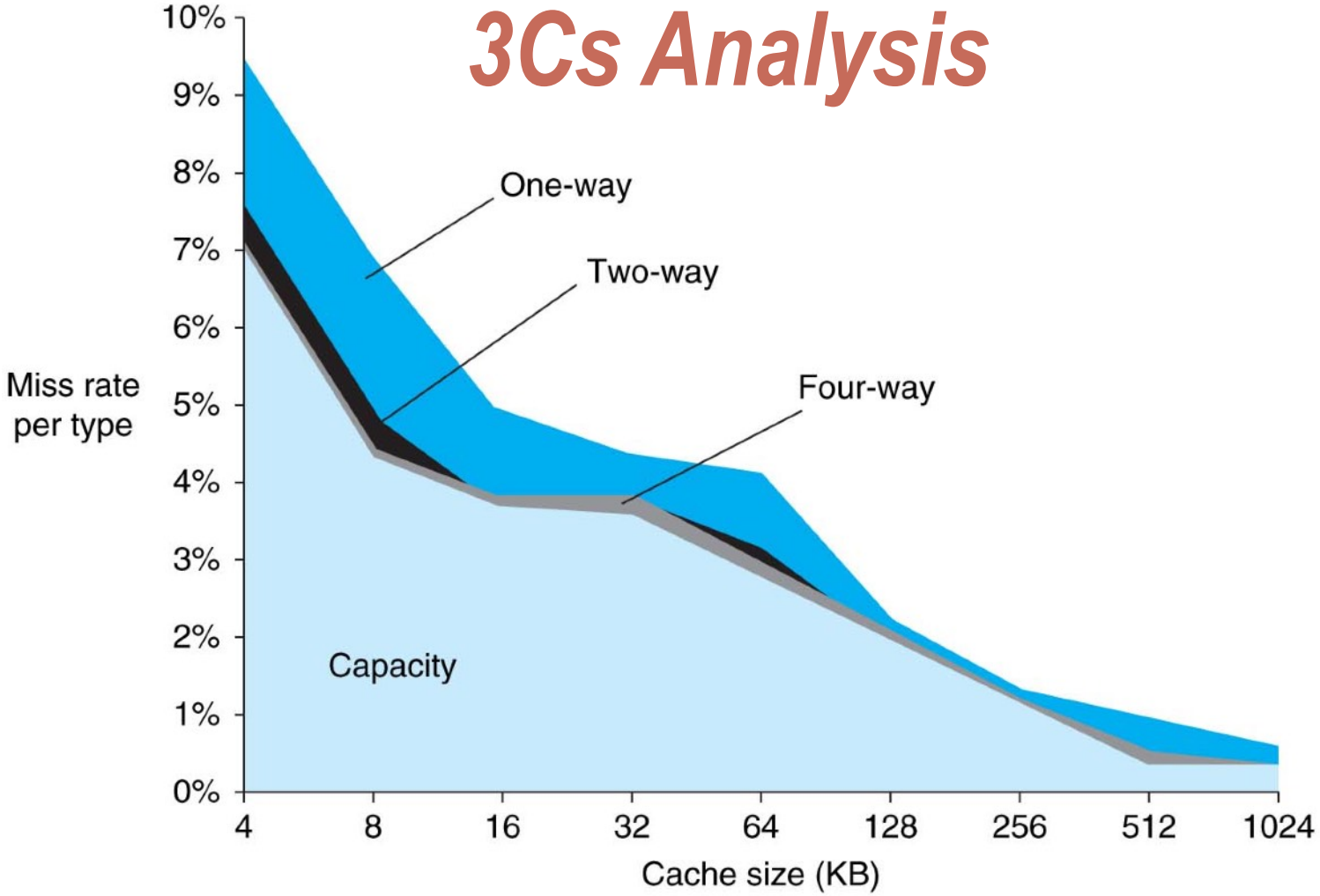
- Multiple memory locations mapped to the same cache location
- Solution 1: increase cache size
- Solution 2: increase associativity

Capacity:

- Cache cannot contain all blocks access by the program
- Solution: increase cache size

Invalidation: other process (e.g., I/O) updates memory

3Cs Analysis



- Three sources of misses (SPEC2000 integer and floating-point benchmarks)
  - Compulsory misses 0.006%; not visible
  - Capacity misses, function of cache size
  - Conflict portion depends on associativity and cache size

4 Questions for Caches (and Memory Hierarchy)

Q1: Where can a block be placed in the upper level?
(Block placement)

Q2: How is a block found if it is in the upper level?
(Block identification)

Q3: Which block should be replaced on a miss?
(Block replacement)

Q4: What happens on a write?
(Write strategy)

Q1: Where can a block be placed in the upper level?

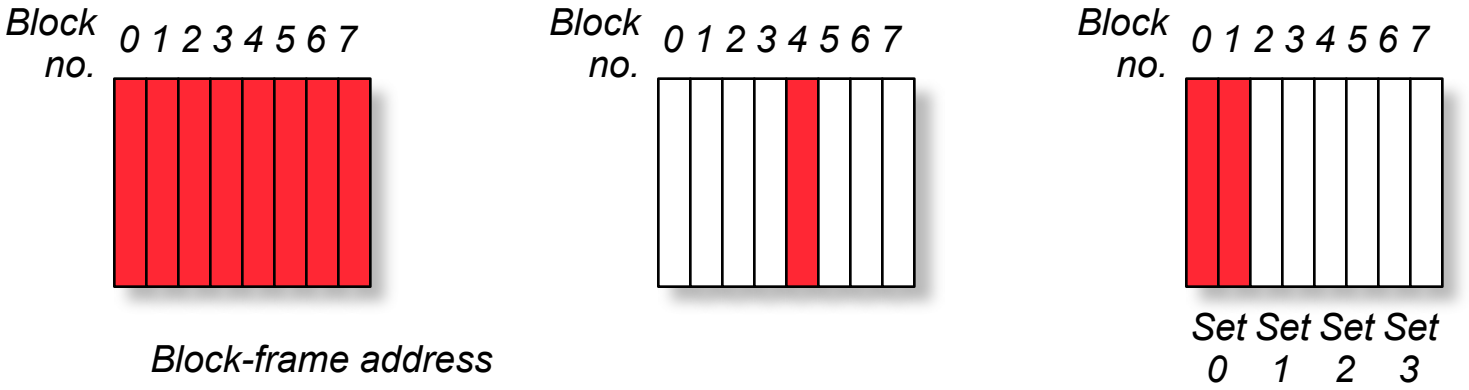
Block 12 placed in 8 block cache:

- **Fully associative, direct mapped, 2-way set associative**
- **S.A. Mapping = Block Number Modulo Number Sets**

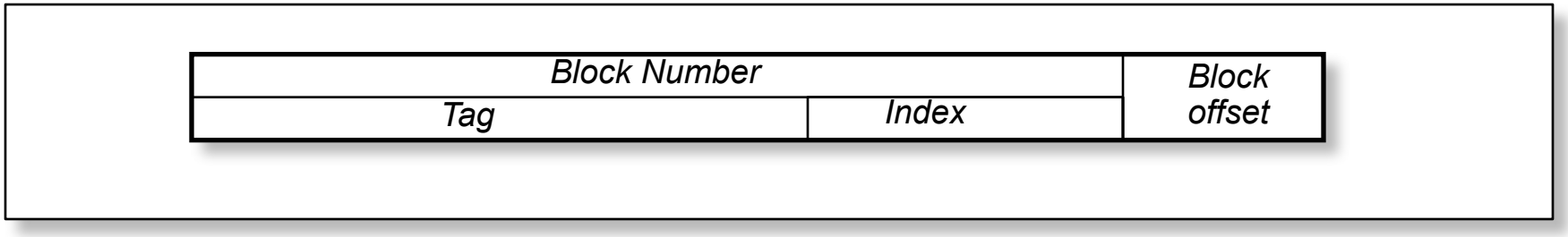
Fully associative: block 12 can go anywhere

Direct mapped: block 12 can go only into block 4 (12 mod 8)

Set associative: block 12 can go anywhere in set 0 (12 mod 4)



Q2: How is a block found if it is in the upper level?



- Direct indexing (using index and block offset), tag compares, or combination
- Increasing associativity shrinks index, expands tag

Q3: Which block should be replaced on a miss?

Easy for Direct Mapped

Set Associative or Fully Associative:

- Random
- LRU (Least Recently Used)

Associativity:	2-way		4-way		8-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Miss-rate

Q4: What happens on a write?

Write through—The information is written to both the block in the cache and to the block in the lower-level memory.

Write back—The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.

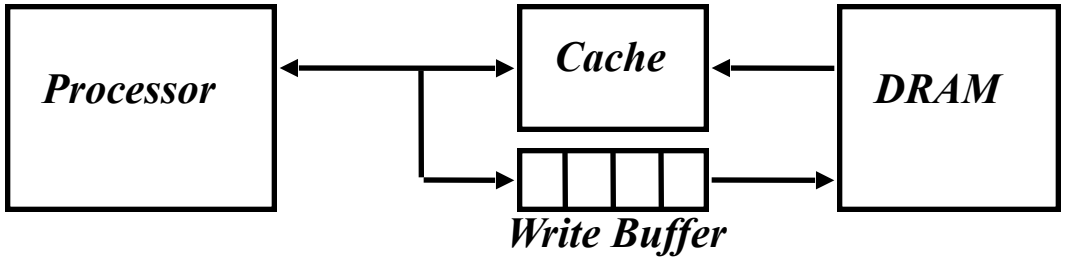
- is block clean or dirty?

Pros and Cons of each?

- WT: simple, read misses cannot result in writes
- WB: no writes of repeated writes (saves energy)

WT always combined with write buffers so that don't wait for lower level memory

Write Buffer for Write Through



A Write Buffer is needed between the Cache and Memory

- Processor: writes data into the cache and the write buffer
- Memory controller: write contents of the buffer to memory

Write buffer is just a FIFO:

- Typical number of entries: 4
- Works fine if: Store frequency (w.r.t. time) << 1 / DRAM write cycle

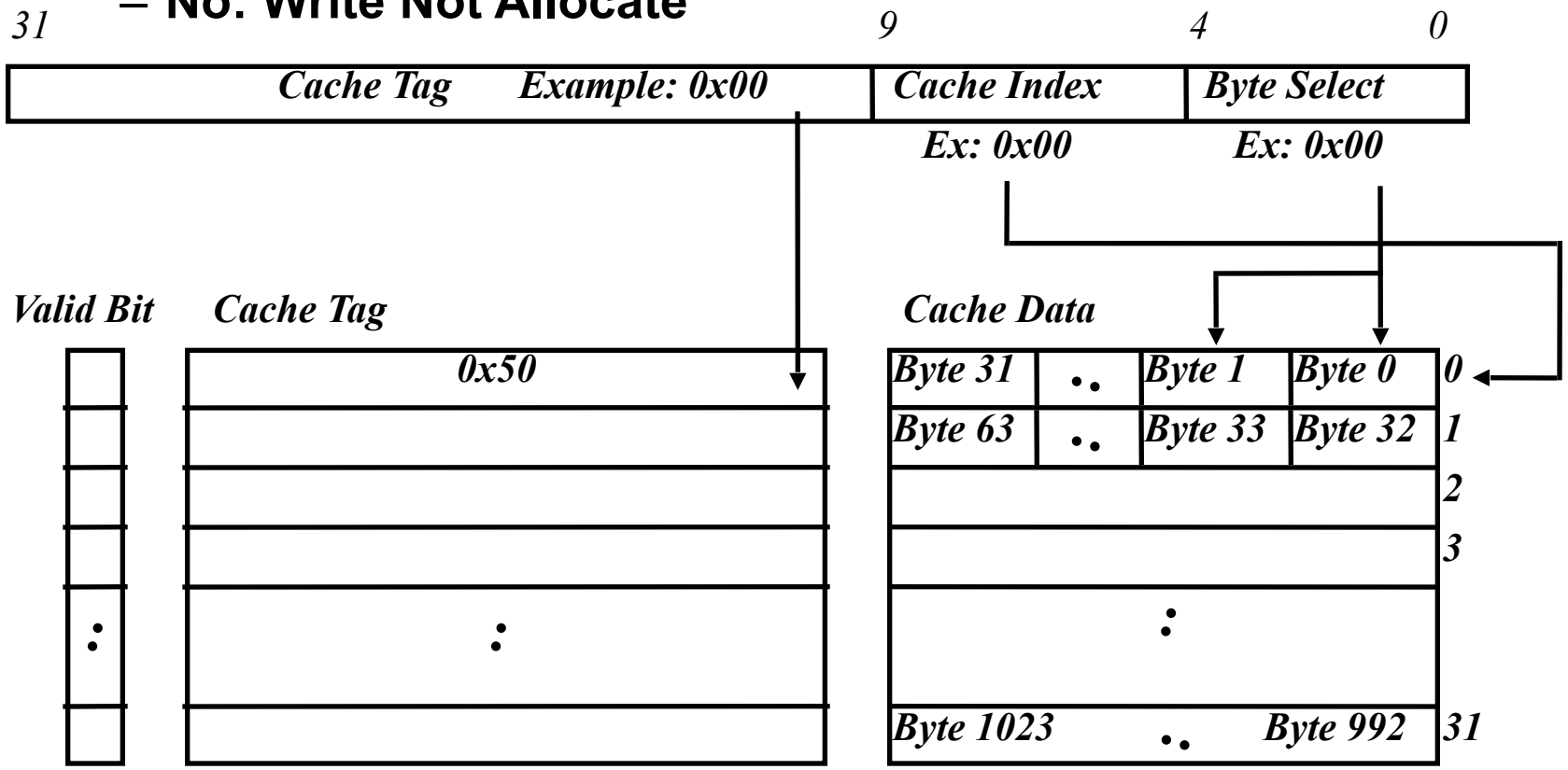
Memory system designer's nightmare:

- Store frequency (w.r.t. time) > 1 / DRAM write cycle
- Write buffer saturation

Write-miss Policy: Write Allocate versus Not Allocate

Assume: a 16-bit write to memory location 0x0 and causes a miss

- **Do we read in the block?**
  - **Yes: Write Allocate**
  - **No: Write Not Allocate**



Usually: Write-back cache uses write allocate.
Write-through cache uses no-write allocate.

Another Real-word Example:

- At ISSCC 2015 in San Francisco, latest IBM mainframe chip details
- z13 designed in 22nm SOI technology with seventeen metal layers, 4 billion transistors/chip
- 8 cores/chip, with 2MB L2 cache, 64MB L3 cache, and 480MB L4 off-chip cache.
- 5GHz clock rate, 6 instructions per cycle, 2 threads/core
- Up to 24 processor chips in shared memory node

IBM z13 Memory Hierarchy